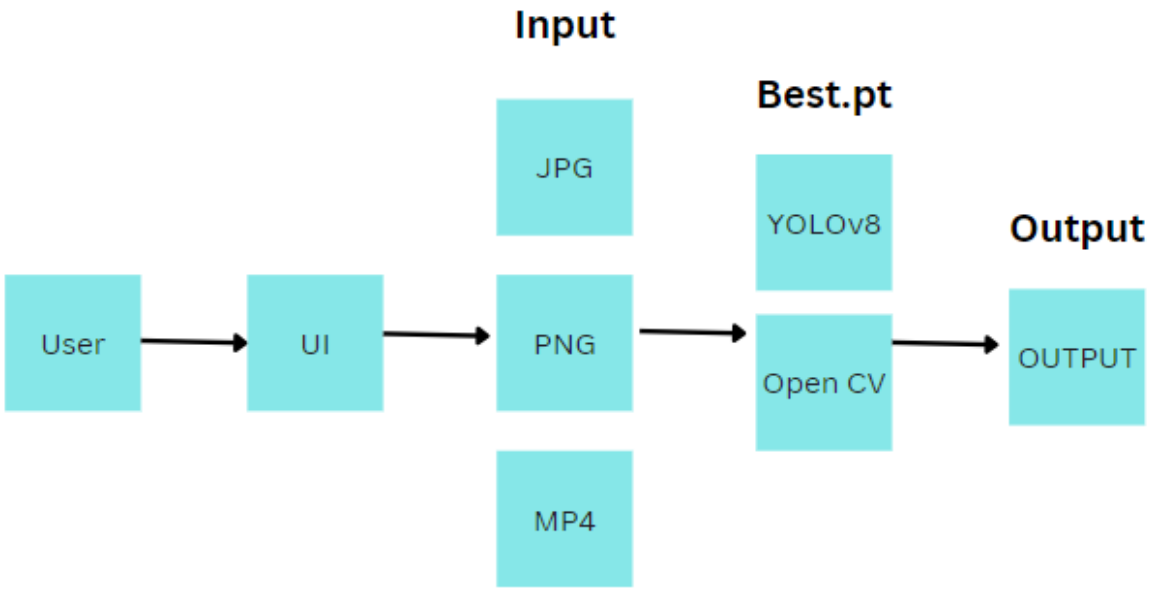
YOLO Weapon detection Project
Team - 593074

Introduction:

A YOLO-based weapon detection project involves using the YOLO (You Only Look Once) algorithm to detect weapons in images or video streams. The YOLO algorithm is a real-time object detection system that is particularly well-suited for detecting multiple objects in an image or video frame. The project involves training a YOLO model on a dataset of images and videos that contain weapons, and then using the trained model to detect weapons in new images or videos. The goal of the project is to provide a tool that can help identify and prevent acts of violence by detecting weapons in real-time, to make our neighborhoods and public spaces safer.

Architecture:



Technical stack used:

Anaconda and Jupyter notebook
Yolov8
Flask: Web framework for building web applications.

Python packages utilized:

- ultralytics: pip install ultralytics
- flask: pip install Flask
- urllib.parse
- os
- cv2
- time

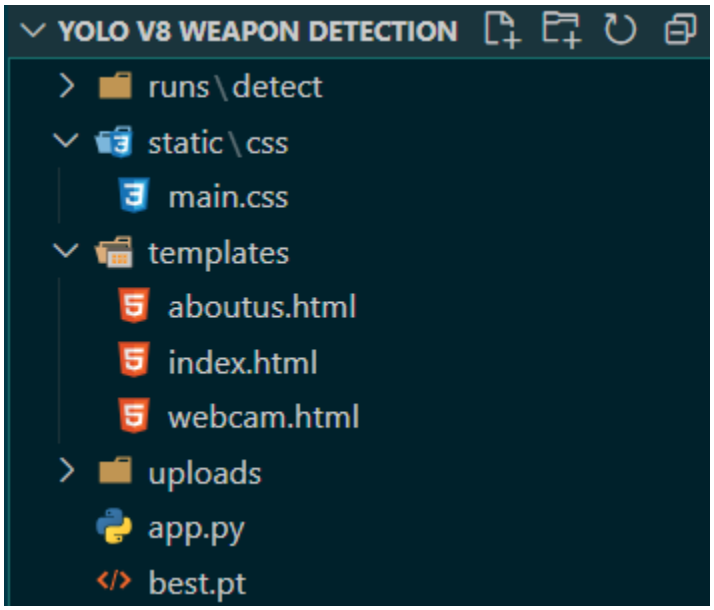
Project objectives:-

- Know how to train the Yolov8 model and
- know how to build a web application using the Flask framework.

Project flow:-

- The user interacts with the UI (User Interface) to choose an image, video, or webcam as the input source.
- The chosen input source (image, video, or webcam feed) is processed by a custom-trained YOLO model integrated with a Flask application.
- The YOLO model analyzes the input, makes predictions, and provides results.
- The predictions are then showcased on the Flask UI.

Project Structure:-



- Uploads has the image and video file we will be testing the web application on.
- We are building a Flask Application that needs HTML pages stored in the templates folder and a python script app.py for server side scripting

- We need the model which is saved and the saved model in this content is best.pt

Milestone 1: Data Collection

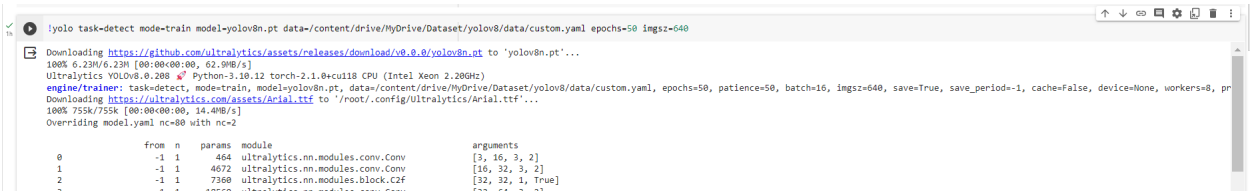
Data was taken from:
https://drive.google.com/drive/folders/1fv9HCfP9DvQwhhcypoO8hbf2xdZI6LL3?usp=share_link

Milestone 2: Model Building

Creating train, test and validation folders

```python
import os
import shutil
from sklearn.model_selection import train_test_split

# Set the path to your dataset
dataset_path = "/content/drive/MyDrive/Dataset/"

# Create YOLOv8 directory
yolov8_path = os.path.join(dataset_path, "yolov8")
os.makedirs(yolov8_path, exist_ok=True)

# Create train, test, and valid directories inside yolov8
sets = ["train", "test", "valid"]
for set_type in sets:
    set_path = os.path.join(yolov8_path, set_type)
    os.makedirs(os.path.join(set_path, "images"), exist_ok=True)
    os.makedirs(os.path.join(set_path, "labels"), exist_ok=True)

# Get the list of image files
image_files = os.listdir(os.path.join(dataset_path, "Images"))

# Split the data into train, test, and valid sets (70:15:15)
train_files, test_valid_files = train_test_split(image_files, test_size=0.3, random_state=42)
test_files, valid_files = train_test_split(test_valid_files, test_size=0.5, random_state=42)

# Move images and labels to the respective YOLOv8 folders
def move_files(file_list, set_type):
    for file in file_list:
        # Move images
        image_src = os.path.join(dataset_path, "Images", file)
        image_dest = os.path.join(yolov8_path, set_type, "images", file)
        shutil.copy(image_src, image_dest)

        # Move labels
        label_file = file.replace(".jpg", ".txt")
        label_src = os.path.join(dataset_path, "Labels", label_file)
        label_dest = os.path.join(yolov8_path, set_type, "labels", label_file)
        shutil.copy(label_src, label_dest)

# Move files to train set
move_files(train_files, "train")

# Move files to test set
move_files(test_files, "test")

# Move files to valid set
move_files(valid_files, "valid")
```

Training



Summary

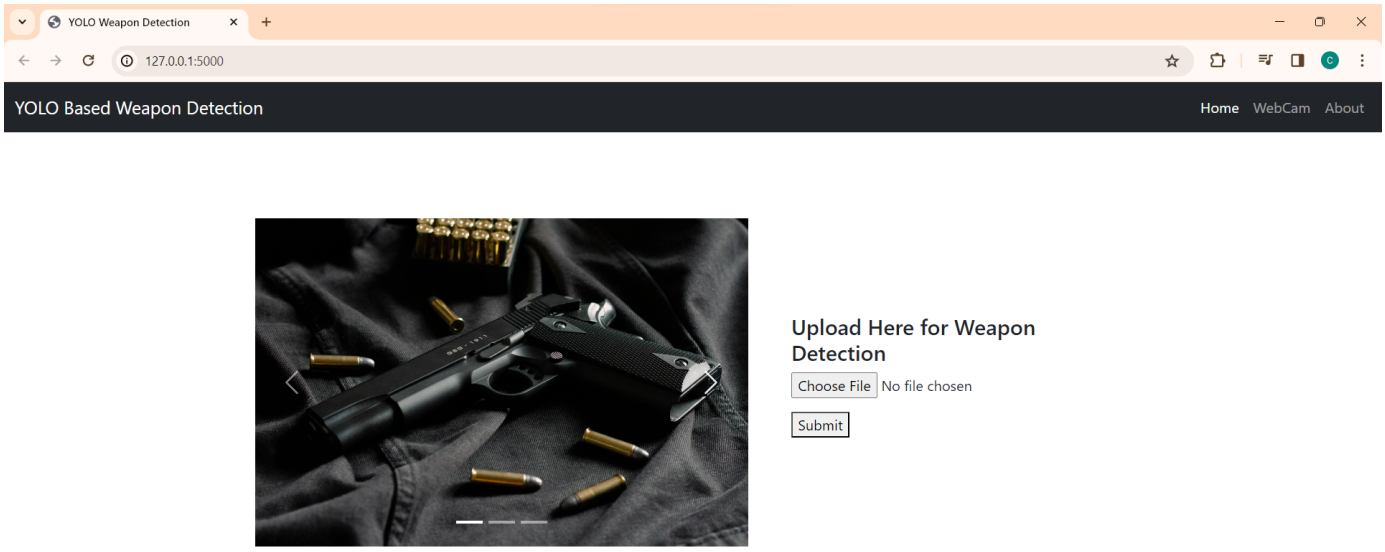
```
Model summary (fused): 168 layers, 3006038 parameters, 0 gradients, 8.1 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:07<00:00,  7.87s/it]
                   all         29         29      0.776      0.674      0.655      0.483
               handgun         29         12      0.748      0.583      0.545      0.441
                 knife         29         17      0.804      0.765      0.765      0.525
Speed: 3.4ms preprocess, 244.2ms inference, 0.0ms loss, 1.7ms postprocess per image
```

Milestone 3: Flask integration

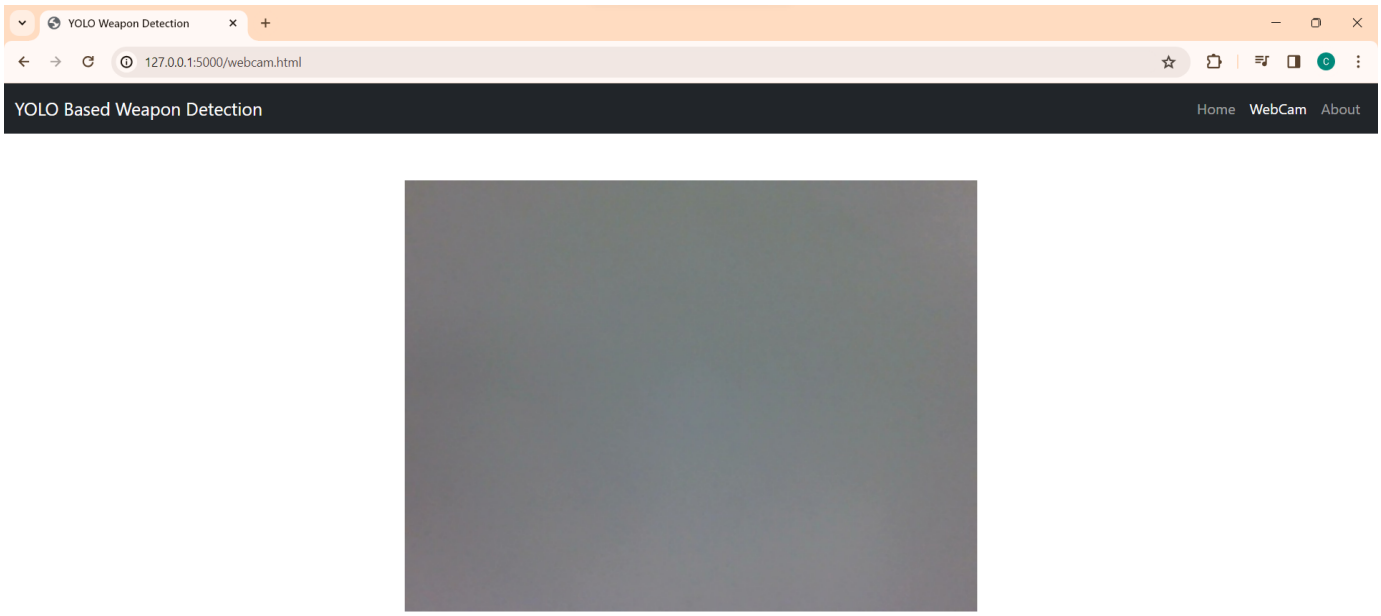
Creating HTML Pages

- We used HTML to create the frontend part of the webpage.
- Here, we have created 3 HTML pages- index.html, webcam.html, and aboutus.html
- Index.html is the home page and also provides facilities of uploading the images and videos for weapon detection.
- webcam.html provides facility to detect weapons with web camera as source.
- We have also used Bootstrap, CSS and Javascript ro enhance the view and functionality of HTML pages.

Home page:

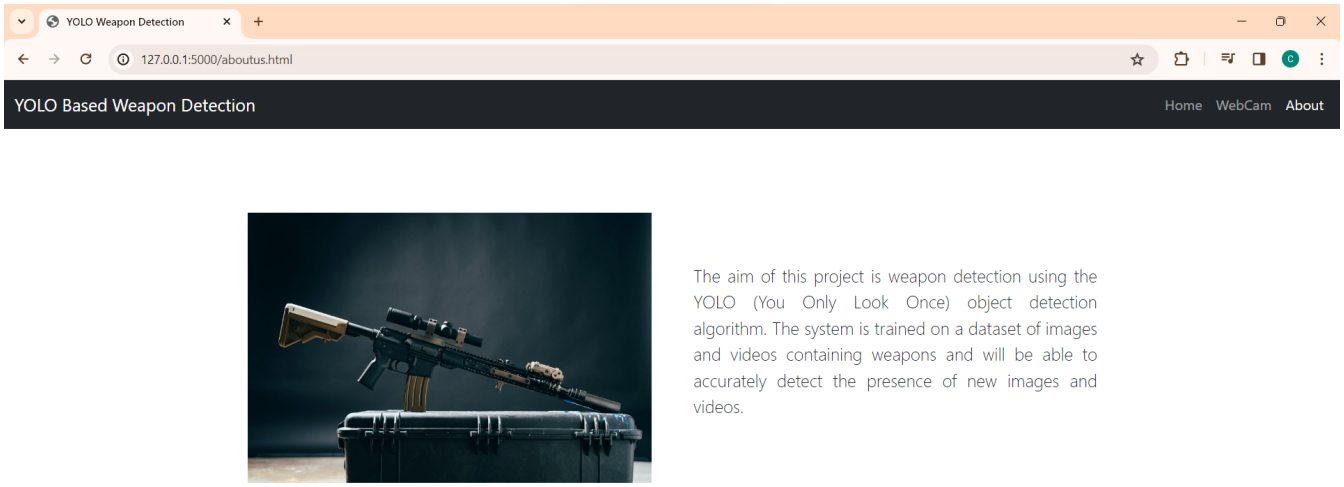


Web camera page:



The screen displays the detection results when it's on.

About page:



Activity 2: Building the python code

Importing Libraries

```python
#Necessary imports
from flask import Flask, render_template, request, send_from_directory, send_file, Response
import os
from urllib.parse import urlparse
from ultralytics import YOLO
import cv2
import time
import math
import winsound
```

app = Flask(__name__)

creates a Flask application object — app — in the current Python module.

```python
app = Flask(__name__)
```

Routing for html pages

index.html

```
@app.route("/")
def index():
    return render_template("index.html")
```

aboutus.html

```
@app.route("/aboutus.html")
def about_us():
    return render_template("aboutus.html")
```

webcam.html

```
@app.route('/webcam.html')
def webcam():
    return render_template('webcam.html')
```

Function for alarm

```
#alarm
def play_alarm():
    duration = 1000
    freq = 440
    winsound.Beep(freq, duration)
```

Statement to start the Flask development server with debugging enabled.

```
if __name__ == "__main__":
    app.run(debug=True)
```

Getting sources via POST

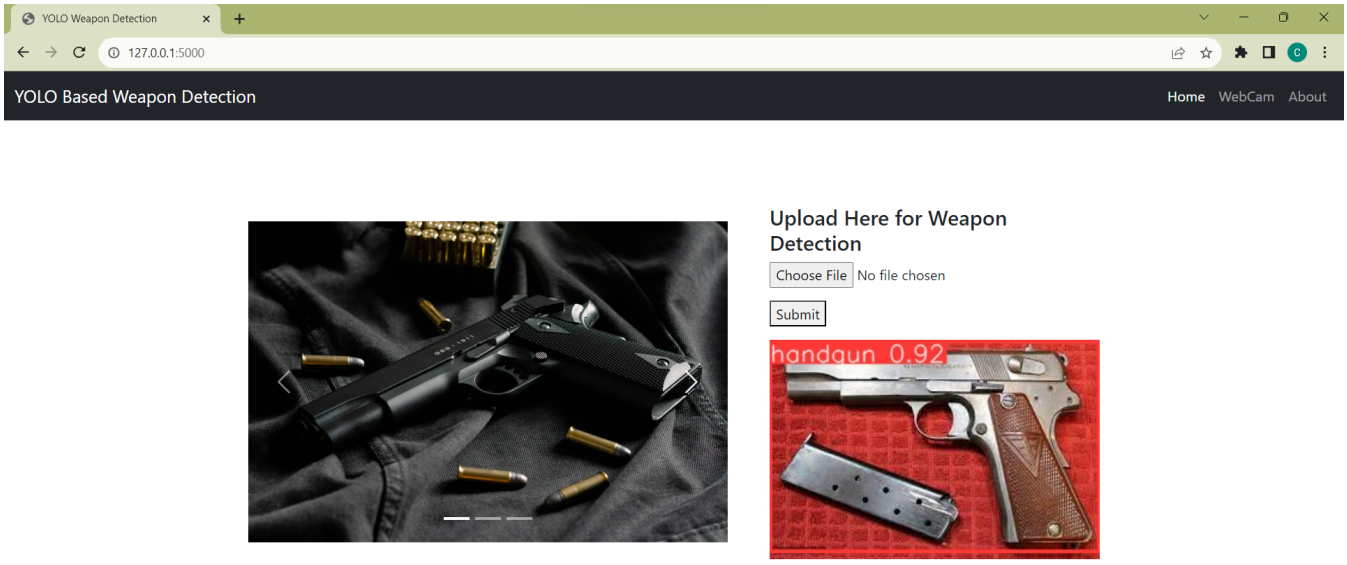
```python
@app.route("/", methods=["GET", "POST"])
def application():
    input_path = None
    alarm_activated = False

    if request.method == "POST":

        if "file_name" in request.files:
            f = request.files["file_name"]
            basepath = os.path.dirname(__file__)
            filepath = os.path.join(basepath, "uploads", f.filename)
            print("Upload folder is ", filepath)
            file_extension = f.filename.rsplit(".", 1)[1]
```

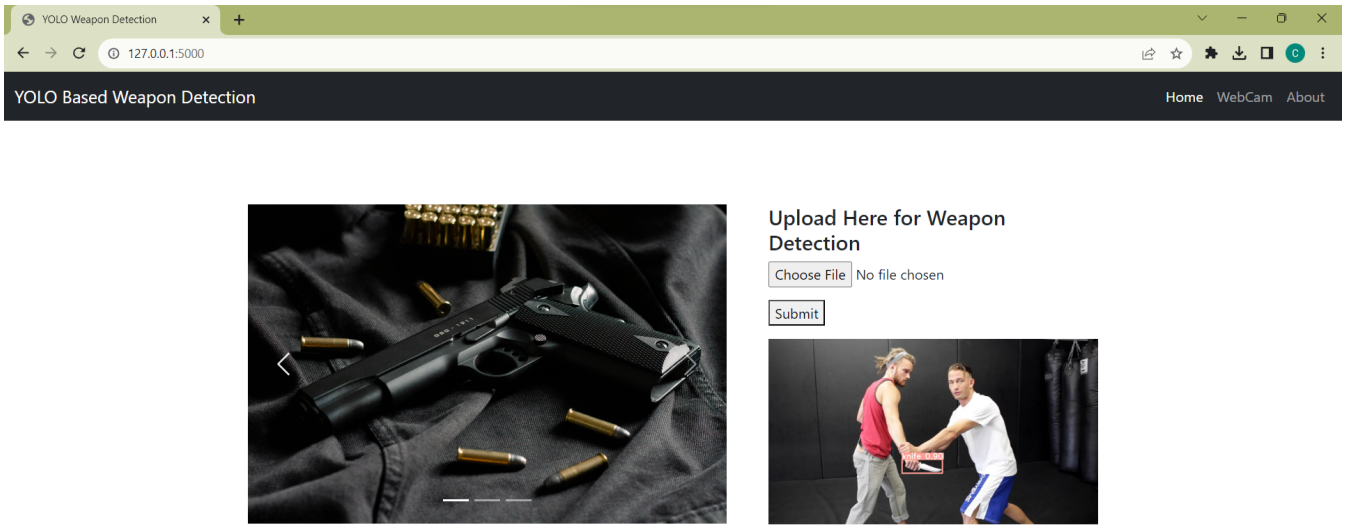
Image as source detection

```python
@app.route("/display/<path:filename>")
def display(filename):
    parsed_url = urlparse(request.url)
    url_path = parsed_url.path
    directory, filename = url_path.split("/display/")[1].rsplit("/", 1)
    input_path = os.path.join(directory, filename).replace("\\", "/")
    print(input_path)
    return send_file(input_path, as_attachment=False)
```

```python
    #Image detection Section
    if file_extension in ("jpg", "png"):
        img=cv2.imread(filepath)
        model=YOLO('best.pt')
        model.predict(source = img, save_txt=True,save=True)
        folder_path = 'runs/detect'
        subfolders = [f for f in os.listdir(folder_path) if os.path.isdir(os.path.join(folder_path, f))]
        latest_subfolder = max(subfolders, key=lambda x: os.path.getctime(os.path.join(folder_path, x)))
        input_path = folder_path+'/'+latest_subfolder+'/'+"image0.jpg"
        #alarm if weapon detected.
        with open(os.path.join(folder_path, latest_subfolder, "labels/image0.txt"), "r") as f:
            for line in f:
                if line.strip():
                    alarm_activated = True
                    break
        if alarm_activated:
            print("Weapon detected")
            play_alarm()
        return render_template('index.html', input_path= input_path,video_format=None)
```

Output:-


YOLO Weapon Detection
127.0.0.1:5000
YOLO Based Weapon Detection
Home
WebCam
About
Upload Here for Weapon Detection
Choose File
No file chosen
Submit
handgun 0.92

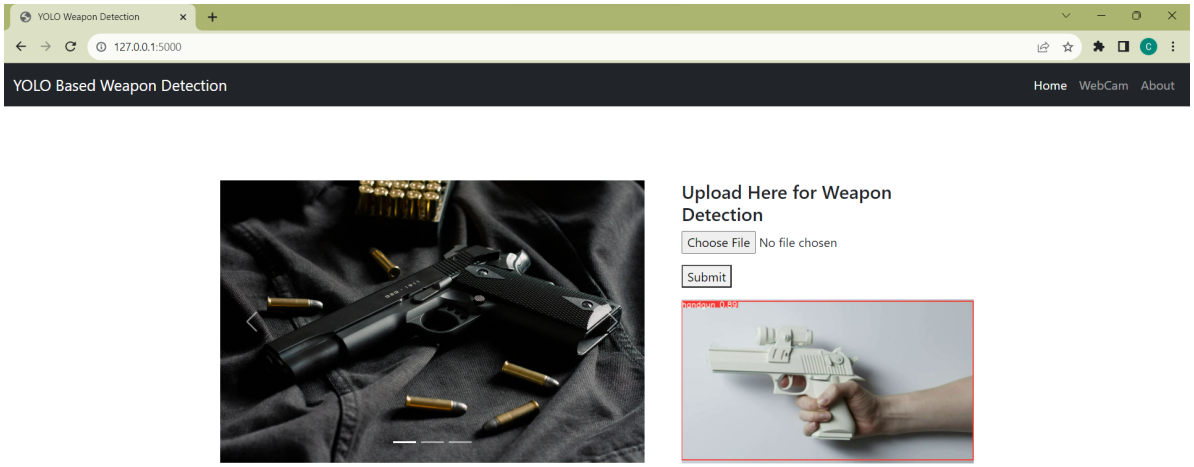

YOLO Weapon Detection
127.0.0.1:5000
YOLO Based Weapon Detection
Home
WebCam
About
Upload Here for Weapon Detection
Choose File
No file chosen
Submit

Video as source detection

```python
@app.route('/video/<path:filename>')
def serve_video(filename):
    video_path = filename
    def generate_frames():
        cap = cv2.VideoCapture(video_path)
        while True:
            success, frame = cap.read()
            if not success:
                break
            ret, buffer = cv2.imencode('.jpg', frame)
            if not ret:
                continue
            frame_bytes = buffer.tobytes()
            yield (b'--frame\r\n'
                   b'Content-Type: image/jpeg\r\n\r\n' + frame_bytes + b'\r\n')
            time.sleep(0.1)
    return Response(generate_frames(), mimetype='multipart/x-mixed-replace; boundary=frame')
```

```python
    #Video Section
    elif file_extension in ("mp4"):
        model=YOLO('best.pt')
        model.predict(source = filepath,save_txt=True,save=True)
        folder_path = 'runs/detect'
        subfolders = [f for f in os.listdir(folder_path) if os.path.isdir(os.path.join(folder_path, f)
        latest_subfolder = max(subfolders, key=lambda x: os.path.getctime(os.path.join(folder_path, x)
        input_path = folder_path+'/'+latest_subfolder+'/'+f.filename.replace(".mp4",".avi")

        #alarm if weapon detected.
        for filename in os.listdir(os.path.join(folder_path, latest_subfolder, "labels/")):
            if filename.endswith(".txt"):
                filepath = os.path.join(folder_path, latest_subfolder, "labels/", filename)
                with open(filepath, "r") as f:
                    # Check for weapon detection
                    for line in f:
                        if line.strip():
                            alarm_activated = True
                            break
                # Play alarm if weapon detected
                if alarm_activated:
                    play_alarm()
                    print("Weapon detected")
                    break  # Break out of the loop once alarm is activated
        if not alarm_activated:
            # No alarm was activated
            print("No weapons detected.")
        return render_template('index.html', input_path= input_path,video_format="mp4")
```

Output:-



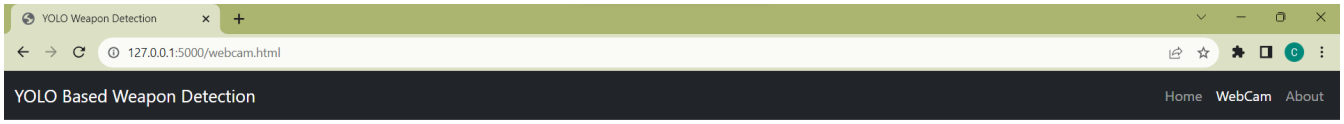
Web Camera as source detection

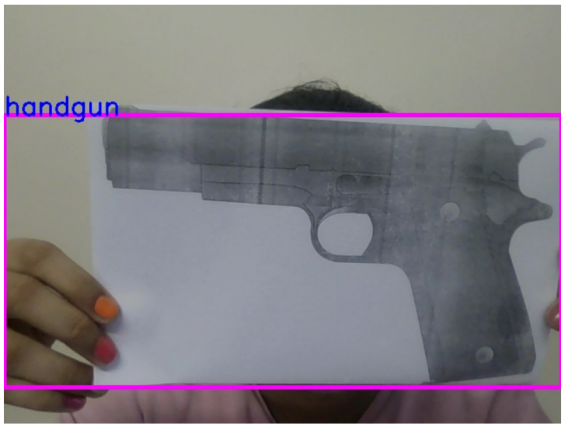
```python
#WebCam section
classNames = ['handgun','knife']
@app.route('/webcam_func')
def webcam_func():
    model=YOLO("best.pt")
    def generate_frames():
        cap = cv2.VideoCapture(0)
        while True:
            success, frame = cap.read()
            results = model(frame, stream=True)
            for r in results:
                boxes = r.boxes
                for box in boxes:
                    x1, y1, x2, y2 = box.xyxy[0]
                    x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2)
                    cv2.rectangle(frame, (x1, y1), (x2, y2), (255, 0, 255), 3)
                    confidence = math.ceil((box.conf[0]*100))/100
                    print("Confidence --->",confidence)
                    cls = int(box.cls[0])
                    print("Class name -->", classNames[cls])
                    org = [x1, y1]
                    font = cv2.FONT_HERSHEY_SIMPLEX
                    fontScale = 1
                    color = (255, 0, 0)
                    thickness = 2
                    cv2.putText(frame, classNames[cls], org, font, fontScale, color, thickness)
                    play_alarm()
            if not success:
                break
            ret, buffer = cv2.imencode('.jpg', frame)
            if not ret:
                continue
            frame_bytes = buffer.tobytes()
            yield (b'--frame\r\n'
                   b'Content-Type: image/jpeg\r\n\r\n' + frame_bytes + b'\r\n')
            time.sleep(0.1)
    return Response(generate_frames(), mimetype='multipart/x-mixed-replace; boundary=frame')
```

```python
@app.route('/webcam_func')
def video_feed():
    return webcam_func()
```

App can be run by python app.py command

Output:-






```
OUTPUT  DEBUG CONSOLE  PROBLEMS  PORTS  TERMINAL

(base)
Charvi Upreti@LAPTOP-ABP6TUMA MINGW64 ~/Desktop/YOLO v8 Weapon detection
$ python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with watchdog (windowsapi)
```